pragma's

product profiles

Issue #34 The *free* newspaper for Pick™ operating system users. April 15, 1987

How to sort by every word in a field

Let's say your company maintains an inventory of parts, and every part has a description. How do you print a catalog that lets you look up any part by any word in any part's description? In other words, imagine a list of parts like:

```
305  WIDE RACHET
467  2" NYLON TUBING
337  BLUE TABLE LEG
288  LARGE RIBBON
```

For an inventory like that, how do we generate a listing sorted by each word in every description, like the following?

```
467  2" NYLON TUBING
337  BLUE TABLE LEG
288  LARGE RIBBON
337  LEG, BLUE TABLE
467  NYLON TUBING, 2"
305  RACHET, WIDE
288  RIBBON, LARGE
337  TABLE LEG, BLUE
467  TUBING, 2" NYLON
305  WIDE RACHET
```

One solution is to read each part description, find a blank in the description, use the position of the blank to split the description into two pieces, save the split description in a temporary file, then repeat the operation for the remaining blanks in the description. Also save a complete copy of the description in the temporary file, as though a split occurred before the first word. After all part descriptions have been split at all blanks, sort the temporary file by the right part of each split description.

For example, the description BLUE TABLE LEG is split twice, once at each blank. If "/" denotes a split, then BLUE/TABLE LEG is the first split description saved in the temporary file, and BLUE TABLE/LEG is the second split description. The complete unsplit description is also saved, to get the effect of /BLUE TABLE LEG.

Assuming that part descriptions are kept in attribute 1 of the parts file and are trimmed of excess blanks, here's a program that reads all parts and saves all split descriptions in a temporary file called TEMP:

```
OPEN "PARTS" TO IN.FILE
  ELSE STOP
OPEN "TEMP" TO OUT.FILE
  ELSE STOP
```

Pragma's Product Profiles
is published periodically by
Semaphore Corporation,
207 Granada Drive,
Aptos, CA 95003,
(408) 688-9200.

Subscriptions are free, but are available only to Pick users with USA mailing addresses.

All material received will be considered for publication. Semaphore reserves the right to edit all submittals.

Semaphore cannot be responsible for lost issues or issues returned to us because of address changes.

Does Your Mailing Label Say RENEW?

If so, your subscription is about to EXPIRE!

To avoid missing your next free issue of *Product Profiles*, renew IMMEDIATELY by filling out and returning the adjacent coupon for a free subscription!

The last two digits of your mailing label number indicate the number of the last issue you'll receive.

Pragma's Product Profiles is completely FREE!

Name ______________________________
Company ______________________________
Address ______________________________
City/State ____________________ ZIP __________
Telephone ______________________________

I use the Pick operating system and my mailing address is in the USA.
Please continue sending me *Pragma's Product Profiles* free of charge.

Signature ____________________ Date __________

Return this form to:
Pragma • 207 Granada Drive • Aptos, CA 95003

Pragma's Product Profiles
207 Granada Drive
Aptos, CA 95003

```
SELECT IN.FILE
10 READNEXT ID ELSE STOP
READV DESC FROM IN.FILE,ID,1
  ELSE GOTO 10
LENGTH=LEN(DESC) ; SPLIT=1
LOOP
  POS=INDEX(DESC," ",SPLIT)
UNTIL POS = 0 DO
  TEMP=DESC[POS+1,LENGTH-POS]
  TEMP<2>=DESC[1,POS-1]
  WRITE TEMP ON
    OUT.FILE, ID:"*":SPLIT
  SPLIT = SPLIT+1
REPEAT
WRITE DESC ON OUT.FILE, ID
GOTO 10
END
```

Since the right part of each split description is kept in attribute 1 of the TEMP items and the left part is in attribute 2, we can define a TEMP dictionary word like

```
    DESC
001 S
002 0
008 A;IF 2="" THEN 1
      ELSE 1:", ":2
009 T
010 50
```

and use the command SORT TEMP BY DESC DESC to get a complete catalog that lists every word from every description. (Since TEMP identifiers are part numbers suffixed by a split count, another correlative is necessary to extract the part number for listing alongside the DESC column.)

Although the DESC definition is an easy way to show split descriptions in one column, they can be difficult to read if part descriptions already contain commas before splitting. A more convenient form of output is a "keyword-in-context" listing, similar to the index published in *Pragma #5*:

```
467              : 2" NYLON TUBING
337              : BLUE TABLE LEG
288              : LARGE RIBBON
337 BLUE TABLE : LEG
467            2" : NYLON TUBING
305          WIDE : RACHET
288         LARGE : RIBBON
337          BLUE : TABLE LEG
467     2" NYLON : TUBING
305              : WIDE RACHET
```

To create a listing like the above, create three more TEMP dictionary words:

```
    LEFT
001 S
002 2
003 \
007 T20
009 R
010 20

    RIGHT
001 S
002 1
003 \
007 T20
009 L
010 20

    CENTER
001 S
002 0
003 \
007 F;":"
009 L
010 1
```

Then use the command SORT TEMP BY RIGHT LEFT CENTER RIGHT to output the keyword-in-context listing. The T20 truncates each description to fit the column without wrapping to another line, while the justification parameter in attribute 9 of LEFT and RIGHT controls which end of a description fragment is truncated. Try using T10 to see exactly how attributes 7 and 9 work together in LEFT and RIGHT.

This month's mailbag

Permanent Profiling?

I enjoyed your article on profiling in issue #33 and was glad to see a well-structured program to perform that task. However, I would like to point out that it probably won't work with programs that are approaching the item size limit of whatever machine you're working on, because the profiler adds too much code.

One way to avoid adding so much code would be to use what could be a permanent form of profiling by incrementing counters at certain selected points. Statement labels are good points to start with.

Instead of incrementing a counter based on line numbers, counters could be referenced by location. Locations can be statement labels or any other identification the programmer wants.

For example:

```
    LOCS = "" ; USED = ""

100 *** PROFILE THIS LABEL ***
    LABEL = "100"
    GOSUB 990 ;*TRACK LABEL 100

990 *** UPDATE LABEL COUNTS ***
    LOCATE(LABEL,LOCS;POS) ELSE
     INS LABEL BEFORE LOCS<POS>
     INS 0 BEFORE USED<POS>
     END
    USED<POS> = USED<POS>+1
    RETURN
```

This method has the disadvantage of probably requiring manual installation, but the profiling code could be left in the program over a longer period of time without having as much impact on performance since monitoring is not done on each line, but on selected points. Additionally, there is no reason to monitor each line of a chunk of code which makes no decisions, since the counts will be the same for each line in the chunk.

David Aitken
Denver CO

The profiler we published adds from 19 to 25 bytes of code to a profiled line, depending on the size of the line number that's used in the STMT<#>=STMT<#>+1 *code, plus 37 bytes for each* STOP *and* ABORT. *Replacing* STMT *with a one-character symbol like* S *will use 6 less bytes per inserted statement. (Remember, it's all right to pick any symbol name you want, since the profiler carefully checks to make sure the inserted code doesn't use a symbol already used by the program being profiled.)*

If your items are reaching a size limit, one alternative is to break them up and use the $INCLUDE *facility now found on many Pick implementations. To let the profiler and profile lister continue to work with* $INCLUDE *segments, you can make them prompt for an offset to be added to every line number in the profiling code. Or, you can get fancy and let the profiler recognize and automatically handle* $INCLUDE *statements, just like the compiler.*

If you don't have $INCLUDE, *one quick and dirty way to temporarily shrink large source code items is to delete all indentations. Another way is to compile with the* (X) *option to create a cross-reference table, then list the table to find the* *variable names taking up the most space in the item. Write a correlative that counts the number of references for each symbol and multiplies by the number of characters in the symbol name. Inevitably, you'll find that a minority of the symbol names will occupy a majority of the item. You can dramatically shrink the item by abbreviating the symbols taking the most space. Don't bother counting single-character symbols, since they can't be abbreviated any further. For example, seven occurrences of the symbol* FILE *requires 28 bytes, while four occurrences of the symbol* LONGNAME *requires 32 bytes. Changing* LONGNAME *to* LGN *everywhere shrinks the item by 20 bytes. Be careful not to abbreviate a symbol down to a name already used elsewhere in the item.*

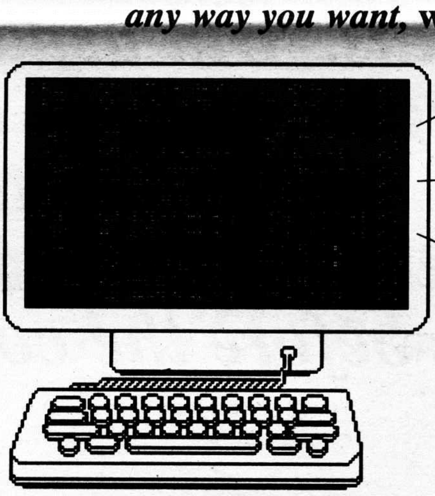
Note that it is very easy to make our profiler only insert code after statement labels and not before every executable statement: change line 27 to simply say GOSUB GET.TOKEN, *and insert* IF TOKEN = ":" *at the beginning of line 28. Lines 23 to 26 and 29 can also be deleted to speed things up. To avoid inserting code on comment lines, just change line 21 to* IF COMMENT THEN RETURN.

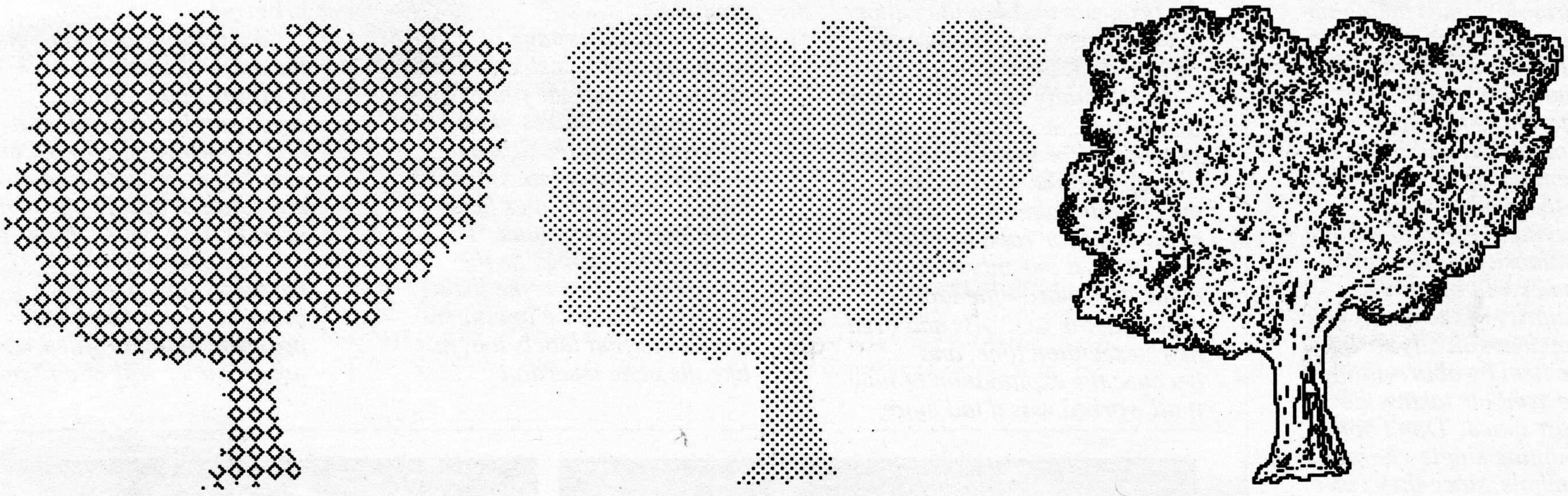
We think permanently leaving in profiling code is a bad idea because it makes programs bigger and harder to maintain and understand, and because it slows down programs far too much, even when only profiling selected points. However, if a compiler offers support for conditional compilation of selected lines, then the runtime overhead can be avoided at the programmer's command, and leaving in profiling code becomes a more attractive option. One "manual" way to have conditional compilation would be to prefix every line of permanent profiling code with something like *PROFILE. *To turn on profiling, make a copy of the program and use the editor to delete every occurrence of* *PROFILE *before compiling.*

We actually considered using GOSUB*s instead of assignment statements for the inserted code, but rejected the idea because they require almost as many extra bytes for each profiled line, at least one more variable name and subroutine label is required, the* GOSUB*s add even more execution time, and because the explanation of how it all worked was a tad more confusing.*

One disadvantage of profiling labels instead of line numbers is that the final profiler output takes more effort to generate. Simply dumping a label table makes it hard to see what lines have what execution counts. To print counts alongside the source code means the listing program has to be intelligent enough to find labels too, just like the code insertion program.

We really can't recommend the manual insertion of profiling code. It's a surprisingly slow, difficult, and laborious job for all but the smallest programs, and not much fun at all. Programmers usually have better things to do. It's easy to make mistakes, and even to introduce bugs into the program being profiled. A programmer will often "know"



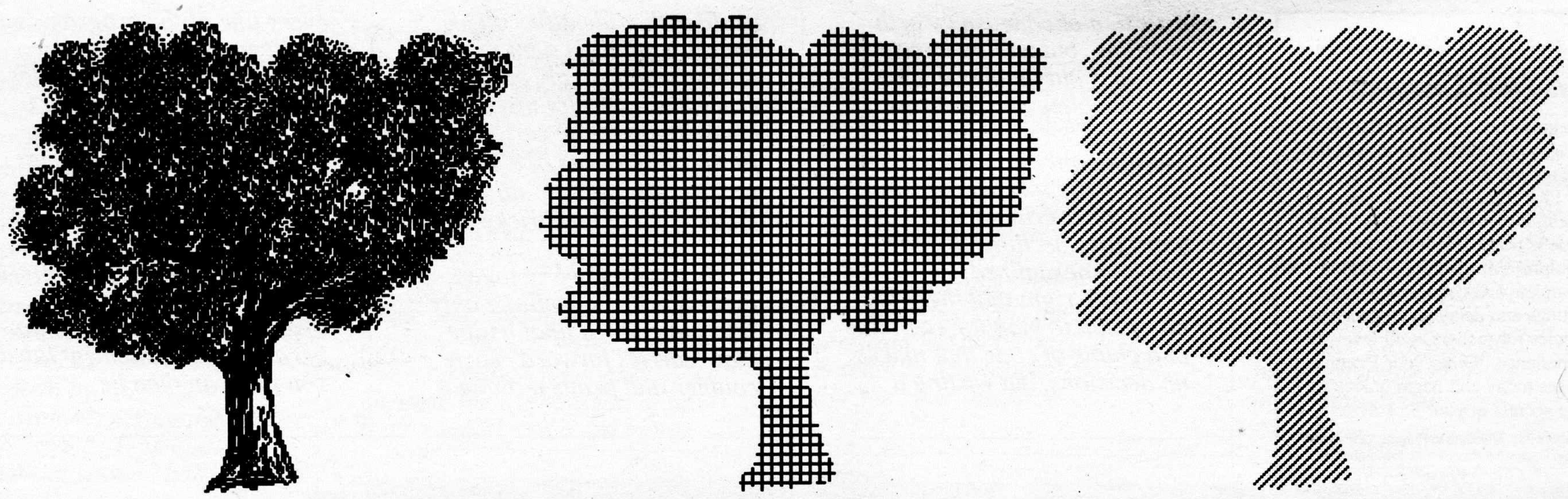


ONCE UPON A TIME, there lived a Pick™ computer user named Mike.

Every workday, from eight to five, Mike would sit in front of a terminal and type commands to make his computer produce reports. Sometimes the commands worked quickly and would make Mike's computer instantly display results. But usually Mike had to type a command that began with the dreaded words SORT or SELECT. Then the computer would take forever to process the command, and Mike would have to wait a long time before the computer could display the report.

Fortunately, Mike could stay busy while waiting for a SORT, because that's when he would always get lots of phone calls from his coworkers, who wanted to know why the displays on their terminals were suddenly slowing down to a crawl.

As each day passed, Mike got more and more bored with his slow computer.



One day, something terrible happened. Mike had just finished waiting ninety minutes for a complicated SORT, and was paging through the report on his terminal. Suddenly, Mike accidentally hit the Return key, and page four of the report flashed by before he could read it. Mike would have to do the whole SORT over again just to see page four. Mike almost had a nervous breakdown.

Fortunately, Mike pulled himself together. But Mike was mad. He just wasn't going to put up with those slow SORTs and SELECTs anymore. So Mike bought B-TREE-P.

Now Mike is very happy. He doesn't have to wait for his computer to SORT anymore. Mike can instantly find and display any data he's looking for. Even scroll forward and backward through his files. And his coworkers no longer call him, because the computer isn't sluggish anymore.

The moral of this story? Buy B-TREE-P.
You'll be as happy as Mike.

B-TREE-P is a product of Semaphore Corporation, 207 Granada Drive, Aptos, CA 95003, (408) 688-9200. Pick is a trademark of Pick Systems.

Unlock The Secrets In Your Computer!

Pragma (not to be confused with *Pragma's Product Profiles*) is the original 48-page technical journal for Pick users published quarterly beginning in August 1982. Each issue is packed with software and helpful information, including complete and debugged program listings and detailed, explanatory articles for readers at all levels of experience. Order your **Pragma** issues today and begin unlocking the secrets in your Pick system!

THIS ISSUE IS SOLD OUT

Pragma #1: Welcome to Pragma ° ZIP Code File Design ° A Program for Renumbering Statement Labels ° Defining [illegible] Counts ° VANILLA, The No-Frills Manufacturing System: The Parts File ° Molectron User Profile ° More Memory, Fewer Reads ° SYSMAP, A Cross-Reference System ° How to Find Missing Checks ° 23 Wish List Items ° Computing Modulos with ENGLISH ° Building ENGLISH Headings with Proc ° 4 Queries ° [illegible] Item Identifiers Hash Well ° Tape Handling ° Patching the Exit Problem in 3.2 SCREENPRO ° The Shell Game.

Pragma #2: We Have Liftoff ° A Modulo Setting Program ° Black Box Formatting ° TRIM.DELIM and PROFILE Utilities ° SYSMAP, A Cross-Reference System: File Format Input ° Galinski Hamburg User Profile ° LOOP vs. LOCATE Benchmarks ° 25 Wish List Items ° An Introduction to ENGLISH: Jargon ° Proc Conversions ° VANILLA, The No-Frills Manufacturing System: Bill of Material Input ° 14 Queries ° The Trouble Tree ° 2 Letters ° A Switchbox for 32 Ports ° Security and the DATA/BASIC Programmer ° The Cookie Game ° Some New Subscribers.

Pragma #3: Is Pragma a Rare Medium, Well Done? ° Edit Aides ° A Proc for Cross-Referencing Q-Pointers ° SYSMAP, A Cross-Reference System: Remaining File Input ° Rainbow Natural Foods User Profile ° More BASIC Benchmark Comparisons ° Justifying Ragged Output ° 13 Wish List Items ° Generating Monthly Column Headings ° VANILLA, The No-Frills Manufacturing System: Purchasing ° 5 Queries ° An Introduction to ENGLISH: More Commands ° Shared Site Checklist ° Converting Paint to Programs ° 3 Letters ° Generating Blank Forms ° Animal, A Game That Learns ° More New Subscribers.

Pragma #4: Survey Says... ° Pacific Valley Bank User Profile ° SYSMAP, A Cross-Reference System: Dicts and Procs ° Uncompiling: Unassembling Stack Code ° Left vs. Right Justification Benchmarks ° 4 Wish List Items ° A Comparison of BASIC Implementations ° More New Subscribers ° An Undocumented Editor Capability ° 3 Local User Group Reports ° Avoiding Saved Lists with PQ-RESELECT ° Self-Documenting Reports ° A Query ° A Survey of Hardware that Supports Pick-Style Software ° Printer Trade-Offs ° An Introduction to ENGLISH: Finding Files ° A Letter ° VANILLA, The No-Frills Manufacturing System: Purchase Order Entry ° The Swat! Game.

Pragma #5: Happy Birthday! ° Interactive Systems Producer Profile ° Uncompiling: Regenerating Source Code ° A Day of Revelation ° IBM Personal Computer vs. Microdata Reality Benchmarks ° VANILLA, The No-Frills Manufacturing System: Receiving ° 3 Wish List Items ° An Introduction to ENGLISH: Being Choosy ° Converting Manual Paint to Programs ° 6 Local User Group Reports ° More New Subscribers ° A Program That Reports File Pointer Locations ° Boiler Plate Processing with Runoff ° A New Query and an Old Query Answered ° SYSMAP, A Cross-Reference System: Automation ° Tape Types ° 6 Letters ° Permuted Index to the First Four Issues of Pragma ° Amazing, a Maze Game.

Pragma #6: Pick Pie Pictured ° The Ubiquitous POINTER-FILE ° Uncompiling: Resolving Labels ° An Introduction to ENGLISH: Syntax Overview ° AccuSoft Producer Profile ° Rounding Out 7 Benchmarks On 5 Machines ° Designing Data Entry Programs ° 12 Wish List Items ° GET: An Input Processor ° 5 Local User Group Reports ° More New Subscribers ° Do You Know Your Proc Limits? ° VANILLA, The No-Frills Manufacturing System: Inspection ° 2 Queries ° Individual Accounts or Shared Accounts? ° Lock Logic Illustrated ° Password Protection ° Two Undocumented Conversions ° A Letter ° How AMAZING Works.

Pragma #7: More New Subscribers ° A New Machine Visits Pragma ° Bantam Hardware Overview ° Suppressing LOCKED Clauses ° An Introduction to ENGLISH: WITH, BY and TOTAL ° Bantam Producer Profile ° VANILLA, The No-Frills Manufacturing System: Disposition ° A Bantam Diary ° New Benchmark Timings for 8 More Machines ° 6 Local User Group Reports ° GET: Source Code ° A Preprocessor for Symbolic Statement Labels ° 2 Wish List Items ° 3 Queries ° Bantam Software Overview ° A Letter ° The Slide Game.

Send $25 for each 48-page back issue to:

Pragma
207 Granada Drive
Aptos, CA 95003

a section of code isn't worth profiling, but will be wrong and miss out on the benefits of a complete and accurate profile. That's why even profiling only statement labels can even be misleading, since many other critical but unprofiled sections of code will go unexamined.

You're right that there is no reason to monitor each line in a chunk of code that makes no decisions, but writing a profiler that identifies all decision points is a bit more difficult than simply profiling every line. Just try inserting counters in all the various forms and layouts of a `LOOP` *or* `IF-THEN-ELSE` *statement for an idea of how tricky the logic can be.*

By the way, we've found that even when profiling only selected code, a final listing that "carries forward" each counter and prints it along every line of the program is much easier to read and understand. That also hides superfluous extra counters accidentally inserted by an imperfect profiling scheme.

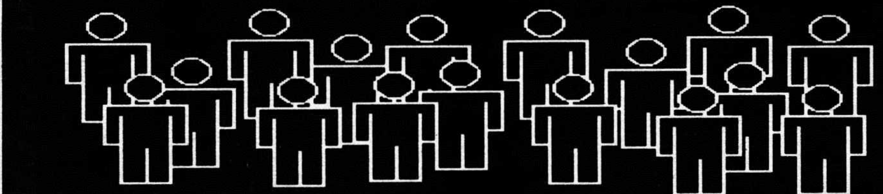
One final note about our profiler: notice how line 62 builds a complete numeric token? If your profiler inserts `GOSUB`*s, building numeric tokens is a good thing to do, because each statement label you find can then be* *compared to the subroutine label you're inserting to make sure you're not adding a label already in use (similar to the test in line 68 to make sure you're not inserting a variable name already in use). But if you're just inserting assignment statements like our profiler already does, then line 62 is unnecessary and can be removed to speed up the parsing of numbers. (Line 62 is an artifact of the original program the profiler is derived from — the renumber utility from Pragma #1 — which has also served as the basis for the Revelation cross reference generator in Pragma #5 and the symbolic label preprocessor in Pragma #7. We have a number of other token processing utilities planned for future issues.) — Editors*

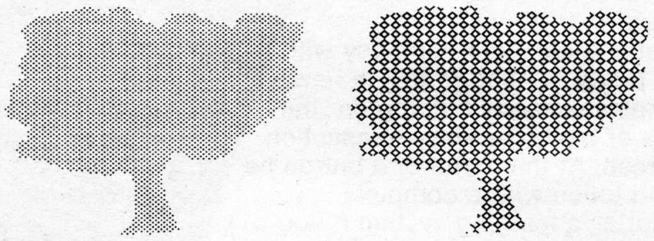
Club notices and newsletters

The following user groups have been regularly mailing us their newsletters and meeting announcements:

California Data Base Management Assoc., Box 3926, Tustin, CA 92681, (619) 471-9307.

Colorado Pick Users, Box





3154, Denver, CO 80201, (303) 321-8569.

MICRU International, 1898 S. Flatiron Ct., Boulder, CO 80301, (303) 444-1645.

National Capital Area Pick Users Group, Box 7594, Silver Spring, MD 20907, (703) 549-20907.

Northern California Pick Users, Box 884474, San Francisco, CA 94188, (415) 284-4063.

Pick Users Group of Arizona, 9248 N. 94th St., Scottsdale, AZ 85258, (602) 860-1122.

Pick Users of Florida, 7000 SW 62nd Ave. #401, S. Miami, FL 33143, (305) 665-0123.

Sierra Pick Information Exchange, Box 19362, Sacramento, CA 95819, (916) 441-7425.

Southern Arizona Pick Users Group, 3130 N. Dodge Blvd., Tucson, AZ 85733, (602) 323-7500.

Richmond Area Pick User Group, Box 3409, Richmond, VA 23235, (804) 358-1714.

Utah Pick Users Group, Box 151275, Salt Lake City, UT 84115, (801) 581-3706.

COMING SOON:

P.U.P.

the Pick Users Pool!

Use C.R. Teacher on Your Pick System to Create MS-DOS Demonstration Diskettes.

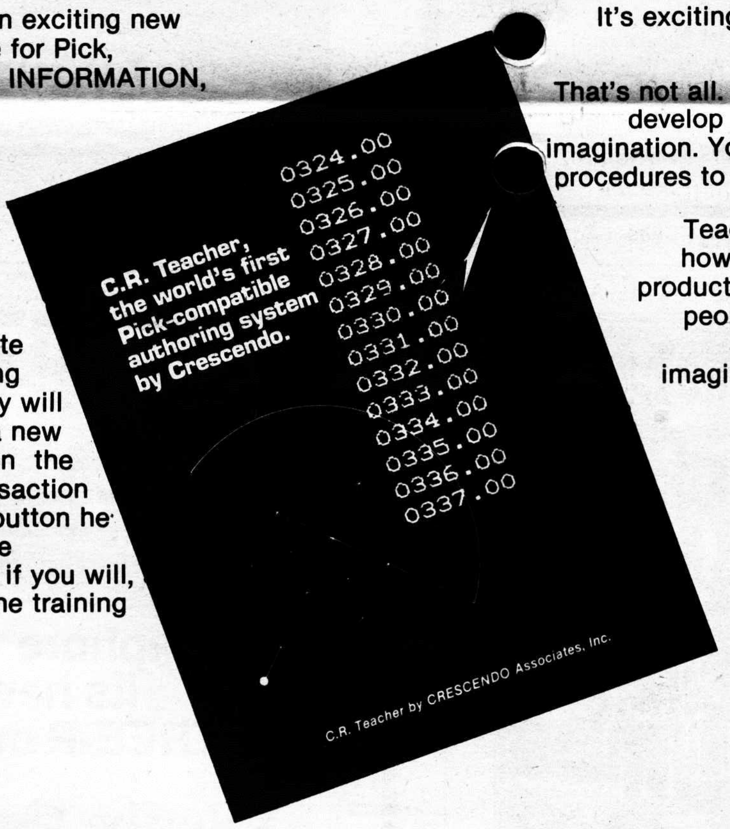
Now *you* can create your own computer-based training and tutorials!

for ■ Sales ■ Manufacturing ■ Office Personnel ■ + other applications limited only by your imagination.

C.R. Teacher software is an exciting new authoring system available for Pick, McDonnell Douglas, Prime INFORMATION, and COSMOS Revelation.

What is an authoring system? It's a tool that YOU can use to create your own computer-based training.

Using C.R. Teacher you possess the power to create the computer-based training that you and your company will be proud of. Picture this: a new employee needs training on the use of the order entry transaction screen. At the touch of a button he is in touch with a complete tutorial; a learning system if you will, which provides him with the training he needs.



C.R. Teacher, the world's first Pick-compatible authoring system by Crescendo.
0324.00
0325.00
0326.00
0327.00
0328.00
0329.00
0330.00
0331.00
0332.00
0333.00
0334.00
0335.00
0336.00
0337.00
C.R. Teacher by CRESCENDO Associates, Inc.

It's exciting and *you* developed it all with C.R. Teacher.

That's not all. The "courseware" you develop is only limited by your imagination. You can teach company procedures to all of your employees.

Teach a new sales person how to sell the company's product line. Teach shop/floor people how the product is assembled. If you can imagine it, you can create it with C.R. Teacher.

C.R. Teacher

Order C.R. Teacher by calling or writing us at:

CRESCENDO Associates, Inc.

24350 Joy Road, Suite 9 Redford Township, Michigan 48239 (313) 537-1919

YES! Please send me more information about C.R. Teacher.

Name ____________________
Company ____________________
Address ____________________
City __________ State _____ Zip _____
Phone (   ) ____________________
Computer System ____________________

Please fill out or attach business card and return.

C.R. Teacher is a registered trademark of Crescendo Associates Inc.
PICK is a registered trademark of PICK Systems, Irvine, California.

Prime INFORMATION is a registered trademark of Prime Computer, Inc.
Revelation is a trademark of COSMOS, Inc.

Do You Know These Companies?

These fifty companies* use a Pick™ computer just like you do, but their computer can do some special things that your computer can't:

Long Beach Community Services • Halprin Supply Co. • Casualty Underwriters Inc. • University of California • Distributed Logic Corp. • Jet Electronics & Technology Inc. • NCAR • Trudell Trailer Sales Inc. • Stewart Co. • Computyme • Data Operating Systems Inc. • Tel-A-Train Inc. • Information Technology Consultants • Life & Health Insurance Co. of America • Flynt Systems Corp. • System Works Inc. • Miami Trading Enterprises • Generation Research • Multisystems Inc. • Infocel • Cooke Data Systems Inc. • John Klein & Assoc. Inc. • Condominium Insurance • Penn Independent Assoc. Inc. • Mark Card • Chandler Lumber Co. • Eye Care & Surgery Center • Chicago Kenworth • Wofford College • Conston Inc. • Assertive Systems • Office Works • Cornell University • City of Irvine • Excalibur Computer Systems Inc. • ADDS Inc. • Specs Music • Specialty Underwriters Inc. • Shoob Photography • May Trucking • Sierra Software Inc. • Medical Accounting Systems • AIPAC • NORPAC • Martin Cadillac Co. Inc. • Capital Software Ltd. • Oman Publishing • Reinsurance Assoc. • Hubert Distributing

Their Pick computers can scroll through files, forward or backward, an item at a time or a page at a time, in any sort order. They don't have to wait for SORTs or SELECTs. They can immediately find any record in any file just by typing one or more starting characters that match any field in the record.

Why are these companies special? Because they purchased B-TREE-P software for using B-trees on Pick computers. B-trees allow any of the data in any of your Pick files to be instantly located, displayed, and processed in any sort order, without having to wait for SORT or SELECT commands.

B-TREE-P and a few minor modifications to your existing data entry programs are all that is necessary for you to immediately be able to search, display, and browse through your data quickly and conveniently. Modifications to your existing data files are absolutely unnecessary! B-trees do not use inverted files, cross-reference tables, or other similar inefficient indexing schemes.

Why not join the ranks of B-TREE-P users who can instantly locate and display their data any way they want, without having to wait for endless SORTs and SELECTs? To order, just send your name, address, telephone number, and your check for $395 payable to Semaphore Corporation at 207 Granada Drive, Aptos, CA 95003. Our phone is (408) 688-9200. We'll send you complete B-TREE-P source code listings and all necessary documentation.

*** This is not an endorsement by any of the companies listed.**

B-TREE-P includes a license agreement with copy, use, and transfer restrictions, limiting your use of B-TREE-P to one computer at a time. Multi-CPU and OEM resale agreements are also available.

Pick is a trademark of Pick Systems.